Memo

To: *Record*
From: *John Gipson*
Date: *January 21, 1999*
Re: *User_Constraint*

This memo describes how to use the *user_constraint* feature of *solve.* This can be used in batch mode by including the line:
USER_CONSTRAINTS complete_path
where complete_path is the complete_path to the program. It can be used in interactive mode by typing "Y" in *optin*.

Theory
Suppose the initial normal equation are

$$NA = B$$

Here A and B are column vectors, and N is the normal matrix. We want to constrain the solution of the normal equation. This is done for a variety of reasons.

1. The normal matrix may not be invertible as it stands, so constraints are necessary to invert it.
2. We may have some a priori knowledge about the solution to the normal equation.
3. The normal equation may be invertible, but the data may still be weak. For example, if we are estimating piecewise linear clocks and atmospheres, and the data is sparse for some period of time, we can strengthen the normal equations by putting constraints on the variation of the clocks and atmospheres. If this is done correctly then we won't change the estimates of the geodetic parameters, although we will decrease their formal errors.

Any constraint has three parts:

1. The constraint vector V.
2. The strength of the constraint, which I will call $1/\sigma^2$.
3. The value of the constraint, which I will call v

One way of looking at constraints is that we bias the solution vector A so that the projection of A along the constraint vector takes the value v

$$V \bullet A \cong v$$

with uncertainty σ. Often the value is 0. For example, in the No-Net-Translation constraints we constrain the sum of the station adjustments for some set of stations to be 0.

To apply a single constraint we modify the normal equations to:

$$\left(N+\frac{1}{\sigma^2}V\otimes V\right)A=B+\frac{v}{\sigma^2}V$$

Here V is the constraint vector, $1/\sigma^2$ is how strong we apply the constraint, and v is the value we are constraining to. As the weight gets larger the constraint dominates the normal equations, and we have

$$\frac{1}{\sigma^2}(V\bullet A)V=\frac{v}{\sigma^2}V\frac{1}{\sigma^2}$$

The solution to this equation is $V\bullet A=v$ with uncertainty σ, which is just what we want.

Practise

The *user_constraints* feature of solve allows the user to apply an arbitrary set of constraints, with arbitrary weights and arbitrary values. All you need to do is write a program which generates the constraints, weights and values. In this section I describe what is involved.

If the user_constraints feature is turned on, then:

1. The program *norml* writes out a file called CNSFxx in the work area. Here xx are the users initials.
   A. The first line of CNSFxx contains *num_parm, 0*, where num_parm is the number of parameters.
   B. The remaining lines are the names of the parameters we are solving for.
2. *norml* schedules the user_constraint program which:
   A. Reads in CNSFxx to find the number of parameters, and their names.
   B. For each constraint it generates the constraint vector, 1/sqrt(wt), and the value. If we want to interpret 1/sqrt(wt) as the sigma then the constraint vector should be normalized to one.
   C. For each constraint this is stored in a (num_parm+2) array: (V,σ,v)
   D. The set of constraint vectors is written out to the file CNSVxx in the work area.
   E. CNSFxx is modifed so that the first line is *num_parm, num_con* where num_con is the number of constraints we want to apply.
3. *norml* then
   A. Reads the CNSFxx to determine the number of constraints.
   B. For each constraint it modifies the normal equations according to the above equation.

In summary, *user_constraint* generates a set of constraint vectors, sigmas, and values:

$$\{V_a\},\ \ \{\sigma_a\},\ \ \{v_a\}$$

where the index a labels the constraint. *Norml* takes these and modifies the normal equations to:

$$\left(N + \sum_a \frac{1}{\sigma_a^2} V_a \otimes V_a \right) A = B + \sum_a \frac{v_a}{\sigma_a^2} V_a$$

The directory /data18/mk3/src/solve/jmg_con contains a number of user_constraint programs. All of these use the routine *usercon.f* plus some other subroutine, for example *hfeopcon.f.* The first of these *usercon.f* takes care of reading and writing the CNSFxx and CNSVxx files. The second file computes the actual constraint.

Example

An example of the above can be found via anonymous ftp in the directory gemini.gsfc.nasa.gov://pub/misc/jmg/xpos_con. This directory contains three files: usercon.f, xposcon.f and makefile. usercon.f takes care of all of the bookkeeping. xposcon.f imposes constraints on the X-component of the station position. The constraint is that the X-adjustment is 100 mm, with strength $\sigma = 1mm$. Of course how close the solution gets to these values depends on how strong the initial data is. The makefile will make the program */home/tmp/XPOSCON.*